UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

| APPLICATION NO. | FILING DATE | FIRST NAMED INVENTOR | ATTORNEY DOCKET NO. | CONFIRMATION NO. |
|---|---|---|---|---|
| 09/894,260 | 06/28/2001 | Richard James Eickemeyer | ROC 9 2000 0282 US1 | 5055 |

7590 04/21/2004

Robert R. Williams
IBM Corporation, Dept. 917
3605 Highway 52 North
Rochester, MN 55901-7829

| EXAMINER | |
|---|---|
| GOLE, AMOL V | |

| ART UNIT | PAPER NUMBER |
|---|---|
| 2183 | 3 |

DATE MAILED: 04/21/2004

Please find below and/or attached an Office communication concerning this application or proceeding.

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

| APPLICATION NO. | FILING DATE | FIRST NAMED INVENTOR | ATTORNEY DOCKET NO. | CONFIRMATION NO. |
|---|---|---|---|---|
| 09/894,260 | 06/28/2001 | Richard James Eickemeyer | ROC 9 2000 0282 US1 | 5055 |

7590 04/21/2004

Robert R. Williams
IBM Corporation, Dept. 917
3605 Highway 52 North
Rochester, MN 55901-7829

| EXAMINER | |
|---|---|
| GOLE, AMOL V | |

| ART UNIT | PAPER NUMBER |
|---|---|
| 2183 | |

DATE MAILED: 04/21/2004

Please find below and/or attached an Office communication concerning this application or proceeding.

| Office Action Summary | Application No. | Applicant(s) | |
|---|---|---|---|
| | 09/894,260 | EICKEMEYER ET AL. | |
| | Examiner | Art Unit | |
| | Amol V. Gole | 2183 | |

*-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --*

**Period for Reply**

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

**Status**

1)☒ Responsive to communication(s) filed on *28 June 2001*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

**Disposition of Claims**

4)☒ Claim(s) *1-15* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-15* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

**Application Papers**

9)☒ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *28 June 2001* is/are: a)☐ accepted or b)☒ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

**Priority under 35 U.S.C. § 119**

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

**Attachment(s)**

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date 2.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

## DETAILED ACTION

1. Claims 1-15 have been examined.

### *Papers Submitted*

2. It is hereby acknowledged that the following papers have been received and placed of record in the file:

#2: IDS (6/28/01)

### *Drawings*

3. The drawings are objected to under 37 CFR 1.83(a). The drawings must show every feature of the invention specified in the claims. Therefore, the branch information queue of claim 5 and 6 must be shown or the feature(s) canceled from the claim(s). No new matter should be entered.

A proposed drawing correction or corrected drawings are required in reply to the Office action to avoid abandonment of the application. The objection to the drawings will not be held in abeyance.

***Specification***

4. The various headings of each section in the specification (e.g. Field of Invention, Background of the Invention, etc.) should not be underlined. Please make the appropriate changes.

5. On pg. 8 line 1, reference is made to an application having serial number 09/64508 entitled *Method for implementing a variable-partitioned queue for simultaneous multithreaded processors.* However it has been identified that the serial number for this application is actually 09/645081. Please make the appropriate changes.

6. On pg. 15, line 21, the figure numbers being referred to are not provided. Please make the appropriate corrections.

7. On pg. 29, lines 10-14, the step 830 as shown in fig. 8 is explained wherein the process checks if the head entry is at the maximum bank number. The steps to be taken when the head entry is at the maximum bank number are provided but no steps are provided for when the head entry is not equal to the maximum bank number (or more appropriately for this invention- when the bank number is less than the maximum bank number). This is not seen in the fig. 8 also. If the step is provided, the Office requests the applicant to show where it is in application or why it is not required.

8. The title of the invention is not descriptive. A new title is required that is clearly indicative of the invention to which the claims are directed.

The following title is suggested: SHARED RESOURCE QUEUE FOR SIMULTANEOUS MULTITHREADING PROCESSING WHEREIN ENTRIES ALLOCATED TO DIFFERENT THREADS ARE CAPABLE OF BEING INTERSPERSED AMONG EACH OTHER AND A HEAD POINTER FOR ONE THREAD IS CAPABLE OF WRAPPING AROUND ITS OWN TAIL IN ORDER TO ACCESS A FREE ENTRY.

***Claim Objections***

9. Claim 5 is objected to because of the following informalities: On line 6 a "/" is required between the "and or". Appropriate correction is required.

***Claim Rejections - 35 USC § 112***

10. The following is a quotation of the second paragraph of 35 U.S.C. 112:

*The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.*

11. Claims **1-10** are rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

12. Claim 1 recites the limitation "the resources of more than one thread" in lines 5-6. There is insufficient antecedent basis for this limitation in the claim.

13. Claim 1 recites the limitation "the entries allocated to one thread" in line 7. There is insufficient antecedent basis for this limitation in the claim.

14. Claim 2 recites the limitation "the last entry" in line 3. There is insufficient antecedent basis for this limitation in the claim.

15. Claim 5 is indefinite because of the use of "and/or" language. The scope of the claims cannot be determined. However for the following art rejection the broadest interpretation will be taken by reading "and/or" as "or".

16. Claim 6 is indefinite because the scope of the claim cannot be determined. The claim calls for "a resource queue... comprising a load reorder queue, a store reorder queue, a global completion table, a branch information queue, at least one of queues comprising:..." It is not clear in light of the specification whether a load reorder queue, a store reorder queue, a global completion table, a branch information queue are part of a resource queue or are they separate queues each being a resource queue.

17. Claim 6 recites the limitation "the entries allocated to one thread" in line 13. There is insufficient antecedent basis for this limitation in the claim.

18. Claim 6 recites the limitation "the resources of more than one thread" in lines 11-12. There is insufficient antecedent basis for this limitation in the claim.

19. Claim 6 recites the limitation "the last entry" in line 17. There is insufficient antecedent basis for this limitation in the claim.

20. Claim 6 recites the limitation "the resources for the at least one thread" in lines 24-25. There is insufficient antecedent basis for this limitation in the claim.

21. Claim 7 recites the limitation "the first entry of a particular thread" in line 5. There is insufficient antecedent basis for this limitation in the claim. It is suggested to change it to "a first entry of said particular thread".

22. Claim 7 recites the limitation "the last entry" in lines 9-10. There is insufficient antecedent basis for this limitation in the claim.

23. Claim 9 recites the limitation "the head pointer" in line 5. There is insufficient antecedent basis for this limitation in the claim.

24. Claim 10 recites the limitation "said issue queues" in line 13. There is insufficient antecedent basis for this limitation in the claim.

***Claim Rejections - 35 USC § 102***

25. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

*A person shall be entitled to a patent unless –*

*(e) the invention was described in (1) an application for patent, published under section 122(b), by another filed in the United States before the invention by the applicant for patent or (2) a patent granted on an application for patent by another filed in the United States before the invention by the applicant for patent, except that an international application filed under the treaty defined in section 351(a) shall have the effects for purposes of this subsection of an application filed in the United States only if the international application designated the*

*United States and was published under Article 21(2) of such treaty in the English language.*

26. Claims **1, 5, 10-12** are rejected under 35 U.S.C. 102(e) as being anticipated by Lee et al. (US006629271B1).

**27. In regard to claim 1:**

28. Lee et al. disclose a resource queue (fig. 2), comprising:

(a) a plurality of entries, each entry having unique resources required for information processing (fig. 2 shows a reorder buffer or ROB 152 with a plurality of entries each having unique resources 72, 74, 76, 78, 80, 82, and 84);

(b) the plurality of entries allocated amongst a plurality of independent hardware threads such that the resources of more than one thread may be within the queue (resource 78 indicates the thread to which the entry belongs i.e. either thread 0 or thread 1. Threads 0 and 1 are hardware threads because they are explicitly visible to the hardware and are processed individually because they have different entries in the ROB 152); and

(c) the entries allocated to one thread being capable of being interspersed among the entries allocated to another thread (fig. 2 shows that the entries for each thread are interspersed).

**29. In regard to claim 5:**

30. Lee et al. teaches the queue of claim 1, wherein the information processing further comprises:

(a) an out-of-order computer processor (fig. 1, 100; col. 5, lines 35-37), and

(b) the resource queue may further comprise a load reorder queue and/or a store reorder queue and/or a global completion table and/or a branch information queue (Lee shows a reorder buffer ROB 152 which is a global completion table because it handles the completion (retirement) of instructions col. 5, lines 3-7).

**31. In regard to claim 10:**

32. Lee et al. discloses a shared resource mechanism in a hardware multithreaded pipeline processor (fig. 1 processor 100, col. 5, lines 15-16), said pipeline processor simultaneously processing a plurality of threads, said shared resource mechanism comprising:

(a) a dispatch stage of said pipeline processor (scheduler 114, col. 3, lines 57-59);

(b) at least one shared resource queue connected to the dispatch stage (fig. 1, ROB 152; fig. 2, element 78 shows that threads 0 and 1 share the queue);

(c) dispatch control logic connected to the dispatch stage and to the at least one shared resource queue (the scheduler 114 receives instructions from the front end 112 when resources are ready [col. 3, lines 57-61] and the ROB stores and deletes instructions [col. 4, lines 66-67; col. 5, lines 6-7]. There must be some control logic for controlling these actions of the scheduler and ROB. Hence, although not shown

explicitly by the Lee reference, it must have dispatch control logic in order to control the scheduler and ROB actions); and

(d) an issue queue of said pipeline processor connected to said dispatch stage and to the at least one shared resource queue (Front End 112 includes an issue queue IQ and is connected to the dispatch stage (scheduler 114) and resource queue (ROB 152) col. 3, lines 39-46, 57-60);

wherein the at least one shared resource queue allocates and deallocates resources (stored [col. 4, lines 66-67] and deleted [col. 5, lines 6-7]) for at least two of said plurality of threads (fig. 2, element 78 shows that threads 0 and 1 share the queue) passing into said issue queue (col. 3, lines 57-60 indicate that instructions are received from the issue queue in the front end 112 of the processor and col. 4, lines 66-67 disclose that the instructions from the front end 112 are sent to the ROB 152. Hence, the resources pass through the issue queue) in response to the dispatch control logic (As explained above, the ROB stores and deletes in response to the inherent control logic).

**33. In regard to claim 11:**

34. Lee et al. discloses an apparatus to enhance processor efficiency, comprising:

(a) means to fetch instructions from a plurality of threads into a hardware multithreaded pipeline processor (col. 5, lines 15-16 disclose that the processor is multithreaded and hence inherently it must have means to fetch instructions from a plurality of threads);

(b) means to distinguish said instructions into one of a plurality of threads (fig. 2 element 78 shows that instructions are distinguished between thread 0 and 1);

(c) means to decode said instructions (col. 3, 40-42);

(d) means to allocate a plurality of entries in at least one shared resource between at least two of the plurality of threads (col. 4, lines 66-67; fig. 2, element 78, col. 5, 14-16);

(e) means to determine if said instructions have sufficient private resources and at least one shared resource queue for dispatching said instructions (col. 3, lines 57-61);

(f) means to dispatch said instructions (col. 3, lines 57-61);

(g) means to deallocate said entries in said at least one shared resource when one of said at least two threads are dispatched (col. 5, lines 3-7 disclose that the entries are deleted from the ROB 152 once the instruction retires and an instruction retires only after it has been properly executed which further requires that the instruction be dispatched into the execution path);

(h) means to execute said instructions and said resources for the one of said at least two threads (col. 4, lines 20-24).

**35. In regard to claim 12:**

36. Lee et al. further discloses the apparatus of claim 11, further comprising:

(a) means to flush the at least one shared resource of all of said entries pertaining to the one of said at least two threads (col. 5, lines 6-7 indicate that

instruction entries are deleted from the ROB 152 once retired. Hence when all the instructions pertaining to one of the said threads retire, the ROB 152 will be flushed of all the corresponding entries).

***Claim Rejections - 35 USC § 103***

37. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> *(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.*

38. Claims **2-4, 6** rejected under 35 U.S.C. 103(a) as being unpatentable over Lee et al. (US006629271B1) in view of Chamdani et al. (US006311261B1) and Yososhima (US20020078317A1).

**39. In regard to claim 2:**

40. Although Lee et al. disclose a shared queue (ROB 152) used to retire instructions in program order, they do not mention that the first entry of one thread is capable of wrapping around the last entry of the same thread.

41. Chamdani et al. teach that the reorder buffer is known to be implemented as a circular FIFO queue (col. 12, lines 10-20) so that instructions can be retired in program order (col. 13, lines 54-58).

42. Yasoshima teaches a circular fifo queue as a shared resource between two distinct data segments A and B (fig. 4; para. 6, para. 27, lines 5-7) wherein the fifo can wrap- around (para. 37, lines 7+). In fig. 4(g), when more data is written to B, the write pointer B, indicating the first entry of B, will advance towards the read pointer B, indicating the last entry of B. The write pointer B is capable of wrapping around the read pointer B in the case when the FIFO B is full i.e. write pointer B is equal to read pointer B as can also be inferred from fig. 1(d). Yasoshima teaches this shared fifo queue efficiently utilizes the fifo memory space (para. 14).

43. One of ordinary skill in the art would have recognized to implement the shared queue (Lee: ROB 152) of Lee as a circular FIFO from the teachings of the Chamdani reference and further using the Yasoshima shared FIFO arrangement by having each thread as a data segment sharing the FIFO to maximize space utilization in the shared queue. Therefore it would have been obvious to one of ordinary skill in the art to have modified the Lee reference by incorporating the features of the Yasoshima shared FIFO into the shared ROB 152 wherein the first entry of one thread is capable of wrapping around the last entry of the same thread.

44. One would have been motivated to do so because the Chamdani reference teaches that the circular FIFO is capable of retiring instruction in program order which is an objective of the Lee reference and the Yasoshima reference teaches a circular FIFO shared by a plurality of data structures which maximizes FIFO space and hence saving costs by not requiring a larger FIFO.

**45. In regard to claim 3:**

46. Although Lee et al. disclose a shared queue (ROB 152) used to retire instructions in program order, they do not teach

(a) a head pointer and a tail pointer for at least one thread wherein the head pointer is the first entry of the at least one thread and the tail pointer is the last entry of the at least one thread, and

(b) one of the unique resources is a bank number to indicate how many times the head pointer has wrapped around the tail pointer in order to maintain an order of the resources for the at least one thread.

47. Chamdani et al. teach that the reorder buffer is known to be implemented as a circular FIFO queue (col. 12, lines 10-20) so that instructions can be retired in program order (col. 13, lines 54-58).

48. Yasoshima teaches a circular FIFO queue as a shared resource between two distinct data segments A and B (fig. 4; para. 6, para. 27, lines 5-7) wherein the fifo can wrap- around (para. 37, lines 7+). Yasoshima further teaches that each data segment has a head pointer (write pointer) and tail pointer (read pointer) wherein the head

pointer is the first entry (next memory location to receive data for FIFO A) of the data structure and the tail pointer is the last entry (next memory location to be read from FIFO A) of the data structure (fig. 4, para. 27). Yasoshima teaches this shared FIFO queue efficiently utilizes the fifo memory space (para. 14).

49. One of ordinary skill in the art would have recognized to implement the shared queue (Lee: ROB 152) of Lee as a circular FIFO from the teachings of the Chamdani reference and further using the Yasoshima shared FIFO arrangement by having each thread as a data segment sharing the FIFO to maximize space utilization in the shared queue. Therefore it would have been obvious to one of ordinary skill in the art to have modified the Lee reference by incorporating the features of the Yasoshima shared FIFO into the shared ROB 152 wherein each thread has a head (write) and tail (read) pointer and the head pointer is the first entry of the at least one thread and the tail pointer is the last entry of each thread.

50. One would have been motivated to do so because the Chamdani reference teaches that the circular FIFO is capable of retiring instruction in program order which is an objective of the Lee reference and the Yasoshima reference teaches a circular FIFO shared by a plurality of data structures which maximizes FIFO space and hence saving costs by not requiring a larger FIFO.

51. However the combination of Lee in view of Chamdani and Yasoshima does not expressly show that one of the unique resources is a bank number to indicate how many times the head pointer has wrapped around the tail pointer in order to maintain an order of the resources for the at least one thread.

52. However this difference is only found in the nonfunctional descriptive material and are not involved in the functioning of the shared circular FIFO queue of the combination. Thus this descriptive material will not distinguish the claimed invention from the prior art in terms of patentability, see *In re Gulack*, 703 F.2d 1381, 1385, 217 USPQ 401, 404 (Fed. Cir. 1983); *In re Lowry*, 32 F.3d 1579, 32 USPQ2d 1031 (Fed. Cir. 1994).

53. Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to store any kind of data as one of the unique resources in the Lee in view of Chamdani and Yasoshima combination because such data does not alter how the queue functions and because the subjective interpretation of the data does not patentably distinguish the claimed invention from prior art. See *Gulack* cited above.

**54. In regard to claim 4:**

55. The combination of Lee in view of Chamdani and Yasoshima teaches the resource queue of claim 3, further comprising:

(a) at least one free pointer for the at least one thread indicating an entry in the queue available for resources of the at least one thread (the write pointer of Yasoshima functions as the free pointer also for each thread because it indicates the next entry in the queue available to receive resources for that thread para. 27, lines 11-16).

**56. In regard to claim 6:**

57. Lee et al. disclose a resource queue (fig. 2, reorder buffer ROB 152) in an out-of-order multithreaded computer processor (fig. 1, 100; col. 5, lines 35-37), comprising:

(a) a load reorder queue (the ROB 152 stores each instruction (col. 4, lines 66-67) which includes load instructions (col. 2, line 49). Hence, ROB 152 acts as a load reorder queue also);

(b) a store reorder queue (the ROB 152 stores each instruction (col. 4, lines 66-67) which includes store instructions (col. 2, line 49). Hence, ROB 152 acts as a store reorder queue also);

(c) a global completion table (ROB 152 is a global completion table because it handles the completion (retirement) of instructions col. 5, lines 3-7);

(d) a branch information queue (ROB 152 stores path information in the path field 80 indicating the which branch path the particular instruction in that entry is in (col. 5, lines 33-41). Hence ROB 152 acts as a branch information queue also),

at least one of the queues comprising:

(i) a plurality of entries, each entry having unique resources required for information processing (fig. 2 shows a reorder buffer or ROB 152 with a plurality of entries each having unique resources 72, 74, 76, 78, 80, 82, and 84);

(ii) the plurality of entries allocated amongst a plurality of independent hardware threads such that the resources of more than one thread may be within the queue (resource 78 indicates the thread to which the entry belongs i.e. either thread 0 or thread 1); and

(iii) the entries allocated to one thread being capable of being interspersed among the entries allocated to another thread (fig. 2 shows that the entries for each thread are interspersed).

58. Although Lee et al. disclose a shared queue (ROB 152) used to retire instructions in program order, they do not teach

a first entry of one thread being capable of wrapping around the last entry of the same thread;

a head pointer and a tail pointer for at least one thread wherein the head pointer is the first entry of the at least one thread and the tail pointer is the last entry of the at least one thread;

a bank number to indicate how many times the head pointer has wrapped around the tail pointer in order to maintain an order of the resources for the at least one thread; and

at least one free pointer for the at least one thread indicating an entry in the queue available for resources of the at least one thread.

59. Chamdani et al. teach that the reorder buffer is known to be implemented as a circular FIFO queue (col. 12, lines 10-20) so that instructions can be retired in program order (col. 13, lines 54-58).

60. Yasoshima teaches a circular fifo queue as a shared resource between two distinct data segments A and B (fig. 4; para. 6, para. 27, lines 5-7) wherein the fifo can wrap- around (para. 37, lines 7+). Yasoshima further teaches that each data segment has a head pointer (write pointer) and tail pointer (read pointer) wherein the head

pointer is the first entry (next memory location to receive data for FIFO A) of the data structure and the tail pointer is the last entry (next memory location to be read from FIFO A) of the data structure (fig. 4, para. 27). Also, in fig. 4(g), when more data is written to B, the write pointer B, indicating the first entry of B, will advance towards the read pointer B, indicating the last entry of B. The write pointer B is capable of wrapping around the read pointer B in the case when the FIFO B is full i.e. write pointer B is equal to read pointer B as can also be inferred from fig. 1(d). Yasoshima teaches this shared fifo queue efficiently utilizes the fifo memory space (para. 14).

61. One of ordinary skill in the art would have recognized to implement the shared queue (Lee: ROB 152) of Lee as a circular FIFO from the teachings of the Chamdani reference and further using the Yasoshima shared FIFO arrangement by having each thread as a data segment sharing the FIFO to maximize space utilization in the shared queue. Therefore it would have been obvious to one of ordinary skill in the art to have modified the Lee reference by incorporating the features of the Yasoshima shared FIFO into the shared ROB 152 wherein the first entry of one thread is capable of wrapping around the last entry of the same thread, each thread has a head (write) and tail (read) pointer and the head pointer is the first entry of the at least one thread and the tail pointer is the last entry of each thread, and at least one free pointer for the at least one thread indicating an entry in the queue available for resources of the at least one thread (the write pointer of Yasoshima functions as the free pointer also for each thread because it indicates the next entry in the queue available to receive resources for that thread para. 27, lines 11-16).

62. One would have been motivated to do so because the Lee reference teaches that the circular FIFO is capable of retiring instruction in program order which is an objective of the Lee reference and the Yasoshima reference teaches a circular FIFO shared by a plurality of data structures which maximizes FIFO space and hence saving costs by not requiring a larger FIFO.

63. However the combination of Lee in view of Chamdani and Yasoshima does not expressly show that one of the unique resources is a bank number to indicate how many times the head pointer has wrapped around the tail pointer in order to maintain an order of the resources for the at least one thread.

64. However this difference is only found in the nonfunctional descriptive material and are not involved in the functioning of the shared circular FIFO queue of the combination. Thus this descriptive material will not distinguish the claimed invention from the prior art in terms of patentability, see In re Gulack, 703 F.2d 1381, 1385, 217 USPQ 401, 404 (Fed. Cir. 1983); In re Lowry, 32 F.3d 1579, 32 USPQ2d 1031 (Fed. Cir. 1994).

65. Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to store any kind of data as one of the unique resources because such data does not alter how the queue functions and because the subjective interpretation of the data does not patentably distinguish the claimed invention.

**66. In regard to claim 6:**

67. Claim 6 can be rejected as above except that if one were to interpret the limitations in claim 6 such that each of the load reorder queue, store reorder queue, global completion table and the branch information queue are indeed separate queues the Lee reference does not teach that limitation as it shows all these queues effectively in the ROB 152 (the ROB 152 stores each instruction (col. 4, lines 66-67) which includes load instructions (col. 2, line 49). Hence, ROB 152 acts as a load reorder queue also; the ROB 152 stores each instruction (col. 4, lines 66-67) which includes stores instructions (col. 2, line 49). Hence, ROB 152 acts as a store reorder queue also; ROB 152 is a global completion table because it handles the completion (retirement) of instructions col. 5, lines 3-7; ROB 152 stores path information in the path field 80 indicating the which branch path the particular instruction in that entry is in (col. 5, lines 33-41). Hence ROB 152 acts as a branch information queue also).

68. "Official Notice" is taken that it is well known and expected in the art to have separate load, store, completion table, and branch information queues so as to have dedicated queues for different instructions and hence faster access.

69. Therefore it would have been obvious to one of ordinary skill in the art at the time of the invention to have separated the queues from the ROB 152 and have a separate load reorder queue, store reorder queue, global completion table, and branch information queue.

70. One of ordinary skill in the art would have been motivated to do so because it would allow for faster access to the queue and therefore improve performance.

71. Claims **7-9** are rejected under 35 U.S.C. 103(a) as being unpatentable over Yasoshima (US 20020078317A1) in view of Lee et al. (US006629271B1).

**72. In regard to claim 7:**

73. Yasoshima discloses a method of allocating a shared resource queue (fig. 3,4; para. 6; para. 27, lines 2-7 indicate that the ring buffer is shared by 2 data segments A and B).

74. Although Yasoshima discloses the steps of:

(a) determining if the shared resource queue is empty for a particular data segment (para. 35, lines 6-9);

(b) finding the first entry of a particular data segment (write pointer for a particular data segment indicates the next location where data will be received i.e. the first entry para. 27, lines 14-16);

(c) determining if the first entry and a free entry of the particular data segment are the same (this feature is deemed inherent to the design because if it is not checked whether the first entry (write pointer) is at a free entry, then an occupied entry may be overwritten and data may be lost);

(d) if not advancing the first entry to the free entry (para. 36, lines 1-3; para. 2, lines 9-13 indicate that the write pointer moves to the next available entry);

(e) incrementing a bank number (a location number in FIFO) if the first entry passes the last entry before it finds the free entry (When the write pointer has passed the last free entry and is increments to the next location number in search for the next

free entry it will be equal to the read pointer and the FIFO will be full [fig. 1(d)] before it finds the next free entry);

(f) allocating the next free entry by storing resources (data; para. 36, lines 1-3) for the particular data segment,

he does not teach these steps for a multithreaded electronic data processing environment.

75. Lee et al. disclose a shared resource queue (fig. 2, ROB 152 has entries for multiple threads) for multithreaded electronic data processing (col. 5, lines 15-16).

76. One of ordinary skill would have recognized that the data segments of Yososhima could be threads of the shared resource queue of Lee and by using the Yososhima FIFO arrangement, the utilization of the shared resource queue of Lee would have been improved (Lee: para. 14). Therefore it would have been obvious to one of ordinary skill in the art at the time of the invention to have modified the Yososhima reference by using it for a multithreaded processing environment.

77. It would have been obvious to do so because the Lee reference teaches that a shared resource queue can shared between threads.

**78. In regard to claim 8:**

79. The combination of Yososhima and Lee et al. teaches the method of claim 7, further comprising deallocating multithreaded resources in the shared resource queue, comprising:

(a) locating the last entry in the shared resource queue pertaining to the particular thread (read pointer for a particular data segment indicates the next location where data will be read from i.e. the last entry para. 27, lines 12-14);

(b) determining if the last entry is also the first entry for the particular thread (para. 35, lines 6-9);

(c) if not, finding the next entry pertaining to the particular thread (fig. 4(d), para. 38, lines 1-3);

(d) determining if the bank number of the next entry is the same as the last entry (para. 35, lines 6-9 inherently disclose that the location number pointed to by the read pointer and the write pointer are the same when there is no data in the FIFO. If the next entry is pointed to by the write pointer then the location number of the next entry is the same as the location number of the read pointer) and if so, deallocating the next entry by marking the resources as invalid (This limitation is deemed inherent because when the read pointer advances, the next entry is read and effectively deallocated from the queue. The resources in that entry are hence invalid because they are not part of the queue anymore); and

(e) if not, then skipping over the next entry and decrementing the bank number (para. 37, lines 7-8 disclose that the FIFO can wrap around. Hence the pointers must be decremented to 0 after reaching the end of the FIFO queue to start from the beginning. Therefore, when at the end of the FIFO queue, the bank number (location number of the entry pointed to) will be decremented to zero and the pointer would skip to the next entry);

(f) finding the next previous entry pertaining to the particular thread (fig. 4(d) and para. 38, lines 1-3 disclose that the read pointer moves to the next entry as data is read out).

**80. In regard to claim 9:**

81. The method of claim 7, further comprising flushing the shared resource queue (reading out of queue para. 38, lines 1-3), comprising the steps of:

(a) setting a flush point indicative of an oldest entry to be deallocated pertaining to the particular thread (the read pointer is indicative of the oldest entry because it points to the next entry to be read col. 4, lines 12-14; hence the read pointer is set as the flush point); and

(b) invalidating all entries between the head pointer (write pointer) and the flush point which have the same and greater bank number than the bank number of the flush point (fig. 4(d) and para. 38 line 1-3 disclose that the read pointer advances towards the write pointer as data is read out and inherently the entries read out are invalidated because they are effectively removed from the queue. Hence all the entries between the read pointer (flush point) and the write pointer will be invalidated (read) and inherently they will have bank (location) numbers greater than or equal to the flush point because they come after the read pointer).

82. Claims **13-15** are rejected under 35 U.S.C. 103(a) as being unpatentable over Lee et al. (US006629271B1).

**83. In regard to claim 13:**

84. Lee et al. discloses a computer processing system, comprising:

(a) a central processing unit (fig. 1 shows the structure of a central processing unit);

(b) a semiconductor memory unit attached to said central processing unit (fig. 1, memory system 119);

(c) at least one memory drive capable of having removable memory (fig. 1, disk memory 105 is external and hence is removable);

(f) a hardware multithreading pipelined processor (col. 5, lines 15-16) within said central processing unit to process at least two independent threads of execution (fig. 2, element 78 shows that ROB 152 of the processor 100 stores entries of two threads 0 and 1 ), said pipelined processor comprising a fetch stage (col. 5, line 40 discloses the function of fetching instructions. Hence the fetch stage is deemed inherent to the design), a decode stage (col. 3, lines 39-43), and a dispatch stage (scheduler 114; col. 3, lines 57-59); and

(g) at least one shared resource queue within said central processing unit, said shared resource queue having a plurality of entries pertaining to more than one thread in which entries pertaining to different threads are interspersed among each other (fig.

1, reorder buffer ROB 152; fig. 2, element 78 shows that threads 0 and 1 share the queue and are interspersed among each other).

85. Lee et al. do not mention

a keyboard/pointing device controller attached to said central processing unit for attachment to a keyboard and/or a pointing device for a user to interact with said computer processing system; and

a plurality of adapters connected to said central processing unit to connect to at least one input/output device for purposes of communicating with other computers, networks, peripheral devices, and display devices;

86. "Official Notice" is taken that it was well known and expected in the art at the time of the invention to have a keyboard/pointing controller for attachment to a keyboard and/or a pointing device for a user to interact with said computer processing system and a plurality of adapters to connect to input/output devices for purposes of communicating with other computers, networks, peripheral devices and display devices attached to the central processing unit.

87. Therefore it would have been obvious to one of ordinary skill at the time of the invention to have modified the Lee et al. reference by adding a keyboard/pointing controller for attachment to a keyboard and/or a pointing device for a user to interact with said computer processing system and a plurality of adapters to connect to input/output devices for purposes of communicating with other computers, networks, peripheral devices and display devices.

**88. In regard to claim 14:**

89. Lee et al. disclose the computer processor of claim 13 wherein a first entry of one thread may be located after a last entry of said one thread (fig. 2 shows that the first entry of thread 0, N, is located after the last entry of thread 0, N+1).

**90. In regard to claim 15:**

91. Lee et al. disclose the computer processor of claim 14, wherein the hardware multithreaded pipelined processor in the central processing unit is an out-of-order processor (col. 5, lines 35-37).

***Conclusion***

92. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure. Applicant is reminded that in amending in reply to a rejection of claims in an application or patent under reexamination, the applicant or patent owner must clearly point out the patentable novelty, which he or she thinks the claims present in view of the state of the art disclosed by the references cited or the objections made. The applicant or patent owner must also show how the amendments avoid such references or objections. See 37 CFR § 1.111.

a. Ng (US005623608A) teaches a circular buffer.

b. Rohlman et al. (US 2001003230) teaches a method of maintaining a circular queue.

c. Buser (US006507921B1) teaches a trace fifo which can overlap the old data.

d. Panwar (US006075931A) teaches efficient FIFO structures.

e. Emer (US 20030105944A1) teaches a FIFO structure with multiple threads (para. 57).

f. Schultheiss (US005465120A) teaches a spiral buffer wherein the oldest data is overwritten by new incoming data.

g. Rodgers et al. (US 20030061258A1) teaches a reorder buffer having resources for 2 threads (fig. 6A).

h. Gulati, M., Bagherzadeh, N. (Performance Study of a Multithreaded Superscalar Microprocessor. Proc. of the 2 nd International Symposium on High-Performance Computer Architectures, February 1996, 291-301) teach scheduling unit shared by threads (sec. 3.2).

i. Ponomarev, D., Kucuk, G., Ghose, K., ("Dynamic Allocation of Datapath Resources for Low Power", in Proc. of Workshop on Complexity-Effective Design, held in conjunction with ISCA-28, June 2001) teach a ROB which can dynamically change its size.

j. Daniele Folegnani , Antonio González, (Energy-effective issue logic, Proceedings of the 28th annual international symposium on Computer

architecture, p.230-239, June 30-July 04, 2001, Göteborg, Sweden) also teach dynamically varying the size of a FIFO buffer.

k. S. Reinhardt and S. Mukherjee (Transient fault detection via simultaneous multithreading. 27th Int'l Symp. on Computer Architecture, June 2000) teach a RUU queue which is shared between threads (fig. 2).

93. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Amol V. Gole whose telephone number is 703-305-8888. The examiner can normally be reached on 9:00-6:30.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Eddie Chan can be reached on 703-305-9712. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

AVG
amol.gole@uspto.gov

EDDIE CHAN
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100